Express Mail Label #: E[illegible]554444US

METHOD AND SYSTEM OF WEIGHTED CONTEXT FEEDBACK FOR RESULT IMPROVEMENT IN INFORMATION RETRIEVAL

CLAIM OF PRIORITY

This application claims the foreign priority benefits under 35 U.S.C. §119 of European application No. 00114341.1 filed on July 4, 2000, which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates in general to the field of document based information retrieval in the intranet and the internet domain, and in particular to a method and system for re-ranking an existing ranked result set of documents.

2. Description of Related Art

Nowadays digital information systems provide easy access to large amounts of information. For example, users can access great quantities of information in databases on a network or even in personal computers. Mere access to large amounts of information has only limited value, however, without knowing what is useful about the information.

Searching for a certain information in a huge amount of almost unstructured information can be a very cumbersome task. Although the probability is relatively high that the desired information is somewhere existent in an existing collection of information and potentially can be found, it is at the same time covered up with all that additional unwanted information.

To retrieve a desired piece of information, there have been proposed several methods. A first approach is to structure raw data before a search is started. Structuring the raw data in advance can be done by introducing a set of taxonomies (like "cars", "health", "computers" etc.) and assigning each retrieved document into one or more of those categories. This can be performed in advance, e.g. by an administrator.

Before a search is started, the user has to preselect one or more categories and thereby reduce the possible amount of returned information. Only the information stored in the selected categories is returned. A simple way to accomplish this is for example having several different indexes for the search engine. The user can then select either one or more indexes as a basis for the search.

The drawback of the above described approach is that extra effort is necessary for preparing the raw data by defining the taxonomy set and by assigning each document into one or more categories (or indexes). Since the information most often is of a dynamic nature, an update of new information or categories (or both) has to be done on a regular basis. Further, there is a certain chance that some information is lost because of a wrong assignment to categories or a missing selection by the user.

A second approach is to structure a search result after having finished the search. Structuring the result of a search is either based on the input the user made when he started the search with some query terms, or an attempt is made to dynamically find similiarities inside of the documents and group (cluster) them together.

The second approach can be implemented by way of clustering the results which means finding "categories" dynamically by looking for similiarities inside of the returned documents. This can be achieved according to several different criteria, for example by scanning for lexical affinities (anyhow related expressions) and bundling those documents that have a certain similiarity. Thereby the total set of returned documents is split into several non-overlapping clusters that contain documents which are assumed to deal with the same or similiar context.

The drawback of the above second approach is that the search engine has no idea which context the user is really looking for. Moreover, the clustering of the documents is performed on the basis of word tuples (a set of search terms) that occur with a certain relation to each other in the documents. Ambiguous terms can cause some documents to be scattered all over the clusters, although from the user point of view they deal with the same context. To find those documents the user has to open and read lots of uninteresting information.

An alternative way to implement the second approach, i.e. structurung the search result after having finished the search, is to sort the returned documents in a descending order, derived by some comparable criterion. This method is commonly known as “Ranking”. The appearance of the search terms in each of the documents is a measurement for the importance of the individual document. All values are normalized so the higher the rank value is, the more importance is assumed.

Various different algorithms are used to determine the individual rank value, most often the document sizes, the index sizes, the total number of returned information or other criteria are considered.

Still another way to implement the second approach is refining the search by adding more precise search terms known as “Narrow Query”. The user starts the search process with an initial query and examines some of the returned documents. For each document he assigns a relevance value that reflects if the appropriate document is of high or low value. The search engine scans the marked documents for terms that occur with a high frequency and uses those terms to synthesize a new query. This new query favors the high frequent terms of the documents with a high relevance and excludes the terms of the documents with a low relevance.

But its drawback is that for good results the user has to examine and mark a lot of documents, otherwise the refined query is more or less random.

The predescribed approaches have common drawback that only the entered search terms can be taken into account. Since search terms often are ambiguous, they can occur in various totally different context and cause a lot of unwanted information to be returned. If, in addition, there are only a few terms entered, there is a high probability that lots of documents get the same high rank value.

Further, in an article by R. Fagin and E. L. Wimmers entitled "Incorporating user preferences in multimedia queries", and published in Proc. 1997 International Conference on Database Theory, pp. 247-261, according to a third approach, it is proposed to weight the search terms in information retrieval with the help of user feedback and to allow to apply weights to any sort of rules. In particular, a formula is described to re-sort and re-rank the result set of a query according to the weighting of the original search terms by the user.

The drawback of the above third approach is that from common experiences in information retrieval, it is known that the typical query consists of very few search terms (1 or 2 terms). In the wast majority of all searches trying to apply any weighting to the search terms would have very little impact. The main reason for the fact that the average search is made with only one or two terms is that the user lacks the conception of what additional context information he needs to enter in order to improve the query and filter the unwanted documents. Following this scenario, the user would have to read some of the returned documents first to obtain some context information which he in turn could then use to create additional query terms. Only with a minimum set of query terms that reflect the desired context, this method would make sense.

Finally, all of the above cited prior art approaches have common drawback that, if the preparation of the raw data as described above is not feasible or in a situation, where information can be lost because of a user error, e.g. by selecting a wrong index, only a method that does a post search improvement is acceptable.

Thereupon, all of the described approaches require a relatively high effort with opening and reading the contents of documents, thereby wasting time with useless information.

SUMMARY OF THE INVENTION

An object of the present invention therefore is to provide an information retrieval method and system which presents only the desired information without wasting a user's time with opening and reading documents of no value.

Another object is to provide a method and a system for ranking information comprised of a set of documents which presents the desired information on top of a ranking hierarchy.

Still another object is to minimize the possibility of user errors, in particular with respect to generating search queries.

The present invention accomplishes the foregoing objectives by gathering context information from the documents, generating at least one rank criterion from the context information and re-ranking the documents, based on the at least one rank criterion.

The concept underlying the present invention is a post search improvement, particularly a human feedback to improve search results. It is emphasized hereby that the proposed re-ranking is only based on the results of a first search procedure.

The criterion for re-ranking hereby consists of terms themselves found in the documents. These terms contain information which can provide feedback to the context of the underlying documents. These terms, designated as "context terms" in the following, are used in a weighting formula in order to separate documents with a matching context from documents with a dismatching context.

The ambiguity of search terms which causes unwanted information to be returned, in a preferred embodiment, is solved by means of context information which is used to filter documents with unwanted context. In another embodiment, a context weighting can be performed on a client data processing system, wherein no iterative and time consuming access to a search engine is necessary.

Using lexical affinities to be evaluated as a criterion for re-ranking, is only exemplary and can be accomplished also by other means that are able to provide a notion of context. A suitable approach could be to use "features" that have been extracted from the documents or to simply create "word statistics". "Features" are names of people, places, companies etc. and can be presented as a context indication very similar to lexical affinities. Especially when several extracted features are grouped and weighted together, the context definition would be very precise. "Word statistics" describe the frequency a given word is occuring within a given document.

In case one finds several different words in a document at least a minimum amount of times - no matter if they occur with a certain proximity to each other like with lexical affinities - and the same is true for another document, it is assumed that both documents are very close to each other concerning their context.

Both information (features as well as word statistics) can also be created by the search engine and transferred to the client together with the result set.

Advantageously, the number of unnecessarily opened documents is hereby essentially reduced since an appropriate weighting of the context terms allows to precise the right context for the desired information.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood more readily from the following detailed description when taken in conjunction with the accompanying drawings, in which:

FIGURE 1 is a combined block and flow diagram depicting a search process with weighted context feedback by lexical affinities, in accordance with the present invention;

FIGURE 2 is a screenshot of a possible implementation of the method according to the present invention; and

FIGURE 3 is a flow diagram depicting an embodiment of the re-ranking mechanism proposed by the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIGURE 1, a combined block and flow diagram is depicted which shows a typical search with weighted context feedback by lexical affinities. The scenario described in the following assumes that it is accepted to start a search with rather unprecise search terms and to try to structure the huge amount of returned data according to context information.

A user 100 starts a search by entering search term(s). The search term(s) is (are) transferred to a search engine 110 which generates a result set 120 ranked by the search term(s). The search engine 110, in parallel, automatically retrieves context information 130 from returned result set 120 which is related 140 to the original set of documents. The context information, in a preferred embodiment, is based on lexical affinities which will be discussed in more detail later.

The search engine 110 presents (displays) 150 the context information 130 to the user 100 and asks for a feedback. The user 100 performs a weighting 160 of the presented context information 130 in a range from “important” to “non-important”.

The result set 120 is then re-ranked 170 with the user-weighted context information 180 to increase the “rank distance” of important and non-important documents. The re-ranking is performed by utilizing a context ranking formula which is described in more detail below. The documents that are now on top of the list (highest context-weighted ranking value) represent the desired information.

Optionally, the re-ranking can be performed iteratively wherein the ranked set of documents ranked by the user-weighted context information 180 is treated as a new set 190 with different weightings, based on the original set, and re-ranked again performing the predescribed steps.

The underlying re-ranking algorithm is based on an adaptation of the prementioned Fagin's and Wimmers' formula. The way to generate context information is the extraction of “lexical affinities”. This method produces pairs of terms that are found in a certain relation within the documents. If one, for example, is searching for the term “display”, that term could occur together with the terms “window” or “show” or “computer” or various other combinations. Each combination is an indication for the context the documents are dealing with.

In the formula of R. Fagin and E. L. Wimmers, the following definitions are assumed:

Let x1, ..., xm be the terms that represent the context information. The function f(x1, ..., xm) is then the ranking function that returns a rank value for a given document with respect to the terms x1, ..., xm. θ = (θ1, ..., θm) is the overall weighting for all of the m terms and θi is the individual weight for term xi. Under the assumption that θ1, ..., θm are all non-negative, $\theta 1 >= \ldots >= \theta m$ and they all sum to one, we use the weighted ranking formula according to Fagin and Wimmers which is:

$$(\theta 1 - \theta 2)\, f(x1) + 2(\theta 2 - \theta 3)\, f(x1, x2) \\ +3(\theta 3 - \theta 4)\, f(x1, x2, x3) + \ldots + m\, \theta m\, f(x1, \ldots, xm) \qquad (1).$$

A scenario where all m context terms get a different rating would increase the necessary calculating effort to the theoretical maximum of m. According to equation (1) we had to calculate the ranking value for each document m times, i.e.

$$f(x1), f(x1,x2), f(x1,x2,x3), \ldots, f(x1,\ldots,xm),$$

since the expressions $(\theta n - \theta n+1)$ in equation (1) would all become non-equal to zero. Besides the enormous visualization effort to allow a user to choose m different ratings for all the m context terms, it seems very unlikely that anybody would really do this. An approach where, for example, three different rating levels are selectable is much more practical.

Let the rating levels be "high", "medium" and "low" and the relation between those three rating levels be:

$$\theta a = 2\, \theta b \text{ and } \theta c = 0.5\, \theta b \qquad (2).$$

During further evaluation with more samples of test documents it might turn out that a stronger spreading of those three levels is more appropriate, such as $\theta a = 4\, \theta b$ and $\theta c = 0.25\, \theta b$ or even more.

Now the rating selections of all of the context terms made by the user can be divided into:

a terms rated with "high"
b terms rated with "medium"
c terms rated with "low" (3)

where all selections sum up to the total number of context terms :

$$a + b + c = m \qquad (4).$$

Because of the assumption that all θ1, ..., θm are non-negative and sum to one, it can be written

$$a\ \theta a + b\ \theta b + c\ \theta c = 1 \text{ and}$$
$$\theta b = (1 - a\ \theta a - c\ \theta c) / b.$$

Because of equation (2) it follows

$$\theta b = (1 - 2a\ \ \theta b - 0.5c\ \theta b) / b$$

and finally

$$\theta b = 1 / (2a + b + 0.5c) \qquad (5).$$

Allowing only three discrete ranking levels, one gets only three groups of θn sorted according to their importance to the user. The first "a" context terms are rated high, the next "b" context terms are rated medium and the last "c" context terms are rated low.

$$\theta 1 = ... = \theta a \text{ and}$$
$$\theta a{+}1 = ... = \theta a{+}b \text{ and}$$
$$\theta a{+}b{+}1 = ... = \theta m \qquad (6).$$

Using equation (6), equation (1) can be simplified into:

$$a(\theta a - \theta a{+}1)\ f(x1,x2, ...,xa)$$
$$+ (a{+}b)(\theta a{+}b - \theta a{+}b{+}1)\ f(x1,x2,...,xa{+}b)$$
$$+ (a{+}b{+}c)\ \theta c\ f(x1,x2,...,xa{+}b{+}c) \qquad (7).$$

In case of the only three relevance values "a", "b", and "c", θa+1 = θb, θa+b = θb and θa+b+1 = θc, one can substitute

$$a(\theta a - \theta b)\, f(x1,x2, \ldots,xa)$$
$$+ (a+b)(\theta b - \theta c)\, f(x1,x2,\ldots,xa+b)$$
$$+ (a+b+c)\, \theta c\, f(x1,x2,\ldots,xa+b+c) \qquad (8).$$

This has to be converted into a normalized form with only dependencies to the number of context terms rated with the three discrete ranking levels and the rank functions of the appropriate context terms.

Following equation (2) one can substitute equation (8) with:

$$a\, \theta b\, f(x1,\ldots,xa)$$
$$+ 0.5(a+b)\, \theta b\, f(x1,\ldots,xa+b)$$
$$+ 0.5(a+b+c)\, \theta b\, f(x1,\ldots,xa+b+c) \qquad (9)$$

and according to equation (5) one finally obtains:

$$[2a\, f(x1,\ldots,xa)$$
$$+ (a+b)\, f(x1,\ldots,xa+b)$$
$$+ (a+b+c)\, f(x1,\ldots,xa+b+c)] / (4a+2b+c) \qquad (10).$$

Equation (10) calculates the relevance of a document with respect to the context terms x1, ..., xm when a, b and c are the number of terms that have been assigned a high (a), medium (b) and low (c) relevance and f(x1, ..., xa), f(x1, ..., xa+b) and f(x1, ..., xa+b+c) are the partial relevance functions of this document with respect to a subset of context terms.

In comparison to equation (1), one now has to call the ranking function only three times with different parameter lists which saves a considerable amount of processing effort. The additional factors in equation (10) are necessary to fulfill the three desiderata of equation (1). For the further details concerning these factors it is referred to the prementioned article by Fagin and Wimmer.

A first way is to feed back the weighting information to the Text Search Engine and calculate the new rank value for all documents of the result set. This reveales some major problems as it is currently not possible to call a ranking function with new parameters (the context terms) on the basis of a prior search, because the statistics of these new parameters are not available. Only the original search terms, that are known during the search process, are complemented by the statistics information of the search indexes and could then be used for a ranking.

Besides this problem, it would also require some additional communication with the search engine that increases the response time to the user.

Therefore, it is proposed that a new and very efficient ranking algorithm to be used solely on the client side without any further access to the Text Search Engine. The processing effort is relatively low, so that it can also be used on so called "thin clients" and a quick response to the user is guaranteed.

The new ranking algorithm is based on the "absolute" ranking of the prior search that was derived from the query terms only. Depending on the occurrence of the context terms in a given document and their importance for the user a new rank value is calculated:

Let Rd be the "absolute" rank value of a given document "d" that resulted from the search, Td = (x1, ..., xn) is the tuple of context terms that are contained in document "d", then is

$$fd(x1, ..., xn) = Rd \quad \text{if } x1, ..., xn \text{ element of } Td$$

and

$$fd(x1, ..., xn) = 0 \quad \text{if } x1, ..., xn \text{ non-element of } Td \qquad (11).$$

This new ranking equation (11) is used together with the adaptation of Fagin's and Wimmer's weighted ranking equation (10) and results in a new rank value for each document. The desired effect is a change in the result set order. Those documents that contain the preferred context terms will appear on top of the list since their new rank value is higher than before. The documents that contain only the context terms that got a low relevance from the user will be pushed to the end of the list since their rank value is lowered.

The above described (re-)ranking algorithm can be implemented in a Java environment with the help of Visual Builders. For the special purpose of weighted context feedback a new Java Bean was developed and seemlessly integrated into the Java Beans component model architecture.

FIGURE 2 depicts a screenshot of a possible implementation of this method. The right side of this screenshot shows the result of a search in a table format. According to the ranking of the result set by the query term (in this case it was the single word "display"), the documents are sorted top-down.

On the left side there is the visualization of the Context Feedback. The left column with the header title "Importance" holds three graphical buttons on each line, the right column with the header title "Context" holds the pairs of lexical affinities on each line. The table is scrollable upward and downward to display all the context information and the corresponding weighting.

Immediately after having performed a search, the right column is filled with all context information that could be extracted from all documents of the result set, the weighting of the individual context information in the left column is all set to medium.

By clicking the plus or minus symbols the weighting for each pair of lexical affinities can be modified. Leaving some entries in the "medium" state is also possible. As soon as the user is done with the weighting, he presses the "Rank new" button and the information is sent to the search client to do a weighted re-rank. This causes the right result set table to refresh its contents with the new ranking order.

FIGURE 3 shows the mechanism that leads to a new ranking value "Rd_new" for each document "d" contained in a result set. As already mentioned, the number of iterations per individual document depends on the number of discrete ranking levels that are offered to the user. Following the example depicted in FIGURE 2 with three different ratings ("high", "medium" and "low") one has to calculate up to three portions per document that make up the new rank value.

For each document it is started with its "absolute" rank value of the unweighted result set "Rd" and set the initial value of "Rd_new" equal to "0", in step 300. Further, it is checked in step 310 whether any of the lexical affinities that have been weighted "high" are contained in document "d". If true 320, the new ranking value "Rd_new" is set to "2a*Rd" in step 330 ("a" is the number of lexical affinities weighted with "high"), otherwise "Rd_new" remains unchanged 340.

In a next iteration 350 one performs a similar calculation for all the lexical affinities that have been rated "high" and "medium", and finally a third iteration 360 takes into account all the "high", "medium" and "low" rated lexical affinities. The last step is a normalization of the new rank value "Rd_new".

It is noted hereby that the above described mechanism for re-ranking documents can be applied to any text documents or non-text documents, i.e. all kinds of documents where a ranking principally can be performed. The term "document" as used herein, should be understood in its broadest sense and means an arbitrary allied amount of information contained in files (as per the Windows operating system usage), documents (as per the MacOS operating system usage), pages (as per the web phraseology usage), and other records, entries or terminology used to describe a unit of a data base, a unit of a file system or a unit of another data collection type, whether or not such units are related or relational.